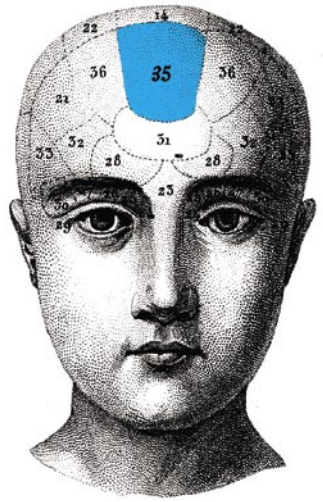
Obrona przed atakami DDoS w Linuksie

Andrzej Nowak, Tomasz Potęga



Nie ma w pełni skutecznych metod zapobiegania atakom Denial of Service. Gdy strumień wrogich pakietów dotrze do celu, pozostaje tylko przeczekać atak. Jedynym sposobem zminimalizowania zagrożeń jest zatrzymanie niechcianego ruchu jak najbliżej miejsca, w którym powstaje.

Atak DoS (*Denial of Service*) to ulubiona broń *script-kiddies* i crackerów. Polega na zalaniu serwera-ofiary taką ilością pakietów, by ten odmówił udostępniania jednej lub wszystkich świadczonych usług. Szczególną odmianą DoS jest technika DDoS (*Distributed Denial of Service*), polegająca na przeprowadzeniu ataku DoS z wielu źródeł jednocześnie.

Wykrycie DDoS rzadko stanowi problem. Jednak ustalenie jego typu jest niemałym wyzwaniem, zaś powstrzymanie dobrze przeprowadzonej napaści graniczy z niemożliwością. Dlatego w przypadku ataków DDoS należy koncentrować się na zwalczaniu objawów, a nie eliminowaniu źródła.

Jak wykryć DDoS

Bogate możliwości filtrowania ruchu sieciowego daje wbudowany w jądro Linuksa firewall, czyli *iptables* (patrz Ramka *Jak działa iptables*). Stwórzmy więc na jego bazie regułę monitorującą i zliczającą przepływ potencjalnie niebezpiecznych pakietów.

Zestaw działających reguł firewalla można wyświetlić za pomocą polecenia `iptables -L` (`--list`). Użycie dodatkowego przełącznika `-v` (`--verbose`) spowoduje wyświetlenie statystyk przepływu pakietów:

```
$ iptables -L -v
Chain INPUT
(policy ACCEPT 14M packets, 6G bytes)
...
Chain FORWARD
(policy ACCEPT 178M packets, 210G bytes)
...
Chain OUTPUT
(policy ACCEPT 12M packets, 4G bytes)
```

Zwróćmy uwagę, że otrzymane statystyki do-

Z artykułu nauczysz się...

- jak wykrywać ataki DDoS,
- jak bronić się przed napaściami tego typu.

Co powinieneś wiedzieć...

- jak skonfigurować, skompilować i zainstalować jądro Linuksa,
- posiadać przynajmniej podstawową wiedzę o linuksowym firewallu – *iptables*,
- czym są ataki *Denial of Service*.

Jak działa iptables

iptables to wbudowany w jądro Linuksa (od serii 2.4) firewall. Umożliwia pełną kontrolę nad przychodzącymi i wychodzącymi z systemu lub sieci pakietami. Jest też doskonałym narzędziem do budowy maskarady NAT (*Network Address Translation*), umożliwiającej udostępnianie połączenia z internetem większej liczbie komputerów.

Firewallem steruje się za pomocą reguł, czyli zestawów poleceń. Podstawowa składnia *iptables* wygląda następująco:

```
iptables -A łańcuch [opcje] -j decyzja
```

Przełącznik `-A` nakazuje firewallowi dodanie reguły do ich obowiązującego zestawu. `Łańcuch` definiuje ruch sieciowy, którego dotyczy reguła. Może on przyjąć następujące wartości:

- `INPUT` – oznacza, że reguła dotyczy filtrowania pakietów przychodzących do komputera, na którym działa *iptables*,
- `OUTPUT` – dotyczy filtrowania ruchu wychodzącego z komputera, na którym działa *iptables*,
- `FORWARD` – odpowiada za filtrowanie ruchu w sieci lokalnej, której udostępniamy internet poprzez NAT. Ten łańcuch umożliwia przekierowywanie lub blokowanie pakietów wychodzących z lokalnej sieci lub do niej trafiających.

Pokazany w ogólnym poleceniu parametr `opcje` pozwala na doprecyzowanie kryteriów, według których mają być filtrowane pakiety:

- `-p protokół` – określa protokół sieciowy, którego dotyczy reguła (na przykład TCP, UDP),
- `-s adres` – określa adres źródłowy objętych regułą pakietów,
- `--sport port [port:port]` – nakazuje stosowanie reguły dla pakietów o podanym porcie źródłowym; umożliwia też wskazanie zakresu portów (np. `--sport 6881:6889`),
- `-d adres` – określa filtrowanie na podstawie docelowego adresu pakietu,
- `--dport port [port:port]` – nakazuje stosowanie reguły dla pakietów o podanym docelowym porcie (portach),
- `-i interfejs` – określa filtrowanie ruchu przychodzącego do firewalla tylko na podany interfejs; tego parametru używa się wyłącznie z łańcuchami INPUT i FORWARD,
- `-o interfejs` – nakazuje filtrowanie wyłącznie pakietów wychodzących z komputera przez podany interfejs sieciowy; ten parametr może być używany tylko z łańcuchem OUTPUT.

Ostatni argument `-j` instruuje firewall, jakie działanie ma wykonać na zdefiniowanych wcześniej pakietach. Argument ten najczęściej występuje z parametrami `ACCEPT` i `DROP`. Jak można się domyślić, parametr `ACCEPT` przepuszcza zdefiniowany pakiet, zaś `DROP` – odrzuca.

Jak to wygląda w praktyce? Załóżmy, że chcemy zablokować wszystkie połączenia komputera o adresie 123.456.78.90 z komputerem, na którym działa *iptables*:

```
iptables -A INPUT -p tcp -s 123.456.78.90 -j DROP
```

Taką regułę najlepiej czytać od końca, czyli: odrzucamy wszystkie połączenia z adresu 123.456.78.90, używające protokołu TCP i przeznaczone dla komputera, na którym działa *iptables*.

Linuksowy firewall ma ogromne możliwości i brak tu miejsca, by je opisać. Więcej szczegółów można znaleźć w artykule *Bezpieczna sieć – zapora ogniowa dla każdego*, zamieszczonym na naszym CD.

tyczą całości ruchu sieciowego. Do wykrycia ataku DDoS potrzebne są tylko informacje o dwóch rodzajach pakietów – ICMP i TCP SYN. W tym celu trzeba założyć nowy łańcuch, na przykład o nazwie *ddos-stats*:

```
$ iptables -N ddos-stats
```

W nim właśnie znajdować się będą liczniki odpowiednich rodzajów pakietów. Najpierw należy dodać pustą regułkę zliczającą wszystkie pakiety:

```
$ iptables -A ddos-stats
```

Następnie rozkażemy *iptables*, by osobno liczył pakiety ICMP:

```
$ iptables -A ddos-stats -p icmp
```

Zaraz potem – pakiety TCP SYN:

```
$ iptables -A ddos-stats -p tcp --syn
```

Charakterystyczny dla ostatnich wywołań jest brak celu łańcucha (`-j`, `--jump`). Reguły takie nie zmieniają tego, co dzieje się z pakietem – powodują jedynie wzrost wartości liczników.

Pozostaje wpiąć nowy łańcuch na początek istniejącego; tego, który jest badany. W tym przypadku będzie to łańcuch FORWARD:

```
$ iptables -I FORWARD -j ddos-stats
```

Teraz, by zebrać interesujące informacje, znowu trzeba wywołać listę reguł *iptables*. Jednak dane zapisane w mega- lub gigabajtach utrudniałyby analizę wyników. Dodatkowy przełącznik `-x` (`--exact`) spowoduje wyświetlenie statystyk natężenia ruchu w bajtach (patrz Listing 1):

Wyciągnięcie wniosków z wyświetlanych w ten sposób statystyk może sprawić pewien kłopot – wymaga samodzielnego obliczenia proporcji podejrzanych pakietów w stosunku do całości ruchu sieciowego. Utrudniłoby to ciągłe monitorowanie zagrożenia atakiem DDoS.

Kontrolę można jednak zautomatyzować za pomocą prostego skryptu. Przykładowe narzędzie zaprezentowano na Listingu 2 (można je też znaleźć na naszym CD). Wywoła ono alarm, gdy stosunek badanych pakietów do całości osiągnie niebezpieczny poziom (w tym wypadku 30 proc.). Warto umieścić skrypt w *crontabie*, aby regularnie kontrolował ilość niebezpiecznych danych.

Listing 2. *Przykładowy skrypt wykrywający atak DdoS*

```
#!/bin/bash
# Przykladowy skrypt badajacy ilosc pakietow

# gdzie jest iptables?
IPTABLES=/sbin/iptables
# łańcuch monitorujący (pierwsza reguła - ICMP, druga - SYN)
STATUS_CHAIN=ddos-stats
# niebezpieczny poziom pakietów danego typu w procentach
HI_LEVEL=30
# co robić, gdy poziomy zostaną przekroczone?
ICMP_ACTION=/bin/true
SYN_ACTION=/bin/true
# czy jest STATUS_CHAIN?
$IPTABLES -L $STATUS_CHAIN -n 2>&1 > /dev/null
if [ $? -gt 0 ];
then
    echo "Nie ma łańcucha $STATUS_CHAIN!"
    exit 1
fi
# łączna liczba pakietów w łańuchu STATUS_CHAIN
TOTAL=`$IPTABLES -L $STATUS_CHAIN -vx | head -3 | tail -1 | awk '{print $1}'`
# ilość pakietów ICMP
ICMP=`$IPTABLES -L $STATUS_CHAIN -vx | head -4 | tail -1 | awk '{print $1}'`
# ilość pakietów SYN
SYN=`$IPTABLES -L $STATUS_CHAIN -vx | head -5 | tail -1 | awk '{print $1}'`
# coś jest, można liczyć...
if [ $TOTAL -gt 0 ];
then
    # policz wartości procentowe
    ICMP_PER=$(($ICMP * 100 / $TOTAL))
    SYN_PER=$(($SYN * 100 / $TOTAL))
    # podaj statystyki
    echo "Razem: $TOTAL"
    echo "ICMP: $ICMP, $ICMP_PER%"
    echo "SYN: $SYN, $SYN_PER%"
    # czy przekroczono poziom pakietów ICMP?
    if [ $ICMP_PER -gt $HI_LEVEL ];
    then
        echo "Duzy udział pakietów ICMP!"
        # dodatkowa akcja
        $ICMP_ACTION
    fi
    # a może SYN?
    if [ $SYN_PER -gt $HI_LEVEL ];
    then
        echo "Duzy udział pakietów SYN!"
        # dodatkowa akcja
        $SYN_ACTION
    fi
else
    echo "Brak pakietów."
fi
# czyścimy liczniki pakietów
$IPTABLES -Z $STATUS_CHAIN 2>&1 > /dev/null
```

Listing 1. *Statystyki natężenia ruchu zbadane za pomocą iptables*

```
$ iptables -L -vx
Chain ddos-stats (1 references)
[...]
  pkts     bytes target  prot opt in   out   source     destination
800386 354292438         all  --  any  any   anywhere   anywhere
    36      2612         icmp --  any  any   anywhere   anywhere
 10070    549460         tcp  --  any  any   anywhere   anywhere
```

Istnieją też oczywiście komercyjne systemy wykrywania tego typu ataków. Zapewne najbardziej znanym produktem jest Peakflow firmy Arbor Networks (*http://www.arbornetworks.com*). DDoS jest jednak najczęściej na tyle agresywny, że wykryje go nawet najprostszy monitor ruchu sieciowego.

Trudna ochrona

Klasyczny atak DoS jest efektem celowych działań i choć zwykle nie przeprowadza się go z prawdziwego adresu nadawcy, ofiara ma duże szanse na wyśledzenie agresora. Inaczej jest z atakami DDoS. Przynajmniej część z nich wykonują tak zwane *zombie* – komputery zainfekowane robakami działającymi w systemach Windows, należące do nieświadomych niczego użytkowników.

Dlatego należy pamiętać o korzystaniu z programów typu IDS (*Intrusion Detection System*), na przykład popularnego narzędzia *snort*. Systemy te wykrywają przede wszystkim sygnatury dostępnych powszechnie pakietów DDoS, a nie same ataki. Maszyny w lokalnej sieci winny też być sprawdzane pod kątem agentów tych pakietów – niewykluczone, ze intruz zdążył już przekształcić nasze komputery w posłuszne *zombie*.

Reverse Path Filtering

Ochrona przed fałszowaniem adresów źródłowych jest najważniejsza przy odpieraniu ataków DDoS. Najprostszy sposób to wykorzystanie informacji o sieciach, do których bezpośrednio podpięty jest komputer. Jeśli interfejs *eth1*, będący bramą dla podsieci 192.168.0/16, otrzyma do przekazania pakiet o adresie spoza tego zakresu, można przyjąć, że jeden z użytkowników ma nie do końca czyste intencje. Filtrowanie na podstawie adresów źródłowych przesyłanych pakietów nazwane zostało *Reverse Path Filtering (RPF)*. Nie jest to panaceum, ale w prostych sieciach sprawdza się znakomicie.

Oczywiście Linux umożliwia korzystanie z RPF. Wszystkich ustawień dokonuje się przez interfejs

/proc. W gałęzi *sys/net/ipv4/conf*, dla komputera z dwiema kartami sieciowymi, znajdują się następujące katalogi:

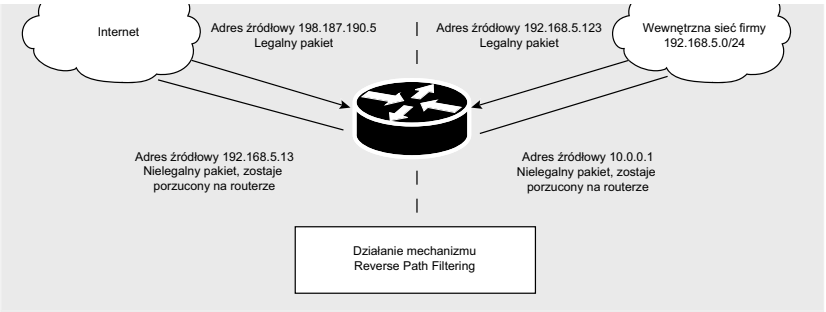
```
$ pwd
/proc/sys/net/ipv4/conf
$ ls
all default eth0 eth1 lo
```

Katalogi *eth0* i *eth1* oraz *lo* zawierają ustawienia dla poszczególnych interfejsów, *default* zawiera wartości nadawane domyślnie, zaś *all* to swego rodzaju globalny przełącznik ustawień.

Przyjrzyjmy się zawartości katalogów (patrz Listing 3). Zwróćmy uwagę na plik *rp_filter*. To właśnie on kontroluje działanie RPF. Wartość niezerowa oznacza włączone filtrowanie:

Listing 3. *Zawartość katalogu /proc/sys/net/ipv4/conf/eht1*

```
$ ls eth1
accept_redirects    disable_policy      log_martians   rp_filter         tag
accept_source_route disable_xfrm        mc_forwarding  secure_redirects
arp_filter          force_igmp_version  medium_id      send_redirects
bootp_relay         forwarding          proxy_arp      shared_media
```



Rysunek 1. *Działanie Reverse Path Filtering*

```
$ cat eth1/rp_filter
0
$ echo 1 > eth1/rp_filter
```

Plik *all/rp_filter* również musi mieć niezerową zawartość. Dopiero wartości niezerowe w obu plikach (*all/rp_filter* i *eth1/rp_filter*) spowodują zastosowanie filtrowania. Dla pewności trzeba jeszcze wyczyścić tablice bufora tras:

```
$ echo 0 > \
  /proc/sys/net/ipv4/route/flush
```

Teraz RPF powinien działać bez zarzutu. Wszystki pakiety pochodzące z fałszywego źródła będą odrzucane.

Standardowo informacje o odrzuconych pakietach nie są zapisywane w logach, ale jeśli do pliku *log_martians* w katalogu *eth1* wpiszemy niezerową wartość, kernel będzie o nich informował. Wymaga to jądra skompilowanego z opcją *IP: verbose route monitoring*, która dostępna staje się przy zaznaczeniu gałęzi *IP: advanced router*. Większość standardowo skompilowanych jąder nie ma włączonej tej opcji, więc trzeba się liczyć z rekompilacją.

Pakiety znikąd

Bogon jest nieformalną nazwą określającą pakiet IP, który przemierza Internet, a pochodzi z niezaalokowanej przestrzeni adresowej. Taka przestrzeń to grupa adresów, które jeszcze nie są w użyciu i czekają na przydzielenie odbiorcom. Niezaalokowane przestrzenie adresowe nazywane są *bogon space*.

Informacjami o alokacji przestrzeni adresowej dysponują organizacje rejestrujące (najważniejsza to IANA – *Internet Assigned Numbers Authority*). Na stronie *http://www.iana.org* znajduje się dokument o nazwie *Internet Protocol v4 Address Space*. Zawiera on informacje o przydziale przestrzeni adresów IP, rozróżnianych po ośmiobitowych prefiksach. Jak te informacje wyglądają? Weźmy wpis dla przedrostka 18:

```
018/8 Jan 94 MIT
```

Widać, że w styczniu roku 1994 adresy rozpoczynające się od oktetu 018 przypisane zostały *Massachusetts Institute of Technology* (MIT). Inaczej wygląda przypisanie prefiksu 173:

```
173/8 Apr 03 IANA - Reserved
```

Ten fragment przestrzeni jest zarezerwowany od kwietnia 2003 r. Nie powinniśmy zezwalać na wymianę pakietów z klientami, których adresy należą do takich właśnie bloków.

Poza adresami nieprzypisanymi są też specjalne klasy adresów. Spójrzmy na wpis odpowiadający przedrostkowi 10:

```
010/8 Jun 95 IANA - Private Use See [RFC 1918]
```

Linia ta odsyła do dokumentu RFC opisującego prywatne przestrzenie adresowe. Zwróćmy jednak uwagę, że wśród klas specjalnych jest też 16 bitowa przestrzeń 192.168, dokument IANA opisuje zaś przestrzenie 24 bitowe. Linia odpowiadająca przedrostkowi 192 wygląda więc następująco:

```
192/8 May 93 Various Registries
```

Filtrowanie bogonów na bazie listy IANA może być przyczyną problemów. W ubiegłym roku Telekomunikacja Polska SA otrzymała pulę adresów 83.x.x.x, która wcześniej należała do przestrzeni adresów niezaalokowanych. W efekcie wielu klientów tej firmy z adresami IP zaczynającymi się od 83. skarżyło się, że ma poważne trudności z dostępem do zasobów światowego Internetu. Powodem kłopotów były nieaktualne konfiguracje filtrów na routerach, mimo że IANA pokazała alokację klasy 83.x.x.x w listopadzie 2003.

Limitowanie pakietów

Jeśli RPF nie zatrzyma pakietów, zostaną one skierowane do komputera-ofiary. Czy istnieje zatem metoda ograniczenia zalewu niebezpiecznych pakietów tak, by zminimalizować ryzyko udanego ataku? Jest kilka możliwości – jedna z nich to ograniczanie pasma przeznaczonego dla ruchu konkretnego typu (np. ICMP, pakiety SYN). Inna, której warto przyjrzeć się dokładniej, to proste limitowanie liczby przepuszczanych pakietów.

Taką zdolność ma wbudowany w jądro firewall, czyli *iptables*. Jeden z jego modułów – *limit* – służy do określenia maksymalnej liczby śledzonych pakietów w danej jednostce czasu. Określić możemy m.in. ilość pakietów na minutę (*x/minute* bądź *x/m*) czy na sekundę (*x/second, x/s*). Spróbujmy zatem ograniczyć liczbę przesyłanych w ciągu sekundy pakietów ICMP do dwudziestu:

```
$ iptables -A łańcuch \
  [-d cel/-o interfejs] \
  -p icmp -m limit \
  ! --limit 20/s -j DROP
```

Aby ograniczyć ruch przekazywany przez nasz komputer, powinniśmy wykorzystać łańcuch FORWARD. Określenie celu może okazać się przydatne, gdy nie chcemy limitować ruchu generowanego wewnątrz naszych sieci, a przesyłanego w świat.

W jaki jednak sposób zidentyfikować źródło ataku? Dostępny w *iptables* moduł LOG umożliwia zapisanie informacji o przekazywanych pakietach. Dzięki nim będziemy w stanie odczytać źródłowy adres IP agresora. Wystarczy zamiast podanej wyżej komendy wpisać polecenia:

```
$ iptables -A łańcuch [...] \
  -p icmp -m limit --limit 20/s \
  -j ACCEPT
$ iptables -A łańcuch [...] \
  -p icmp -m limit --limit 2/m \
  -j LOG --log-prefix \
  "*** ICMP flood *** "
$ iptables -A łańcuch [...] \
  -p icmp -j DROP
```

log-prefix to przedrostek, którym dla ułatwienia identyfikacji pakietów oznaczone zostaną wpisy. Efektem będzie taki zapis w logu kernela:

```
*** ICMP flood *** IN=eth0 OUT=eth1
SRC=adres1 DST=adres2 LEN=84 TOS=0x00
PREC=0x00 TTL=55 ID=437 PROTO=ICMP
TYPE=0 CODE=0 ID=14088 SEQ=38
```

Jak widać, moduł LOG również może być limitowany. W przeciwnym wypadku każdy porzucony pakiet zostałby odnotowany w dziennikach systemowych, a to mogłoby grozić zapełnieniem całej przestrzeni dyskowej.

Bogon filtering

Filtrowanie *Reverse Path* to niezbędne minimum. Dla zwiększenia bezpieczeństwa należy również stosować tak zwane *bogon filtering*, czyli blokowanie pakietów nadchodzących z niezaalokowanych adresów IP (patrz Ramka *Pakiety znikąd*).

Wiele firm i dostawców usług internetowych decyduje się nie przepuszczać przez swoje routery pakietów, które zmierzają pod adresy z *bogon space* lub z nich wychodzą. Skąd jednak wiedzieć, jakie adresy należy blokować? Dość popularna metoda prób i błędów – uznawanie nieznanych klas adresów za podejrzane – nie jest dobrym pomysłem. Nie ma gwarancji, że użytkownik nagle nie zechce skorzystać z zasobów sieci, o których po prostu nie wiedzieliśmy.

Musielibyśmy więc sami poszukać, komu przypisano konkretne zakresy adresów. By oszczędzić administratorom sieci poszukiwań, strona *http://www.cymru.com/Documents/*

Listing 4. *Skrypt przetwarzający listy bogonów w formacie CIDR na reguły iptables*

```
#!/usr/bin/perl
#
# This script is used to convert list of ip blocks in cidr format into
# script that can be run to setup linux firewall to filter those blocks
# Script is written by William Leibzon for Completewhois Bogons Project:
# http://www.completewhois.com/bogons/
#
# $1 - should be list of ip blocks in cidr format

$iptables="/sbin/iptables";
$chainname="bogons";
$chainrule="REJECT";

$cidr_filename=@ARGV[0];

if ($cidr_filename eq "") {
  print "Usage: cidr2iptables cidr_list_file\n";
  exit;
}
open ($cidr_fh, $cidr_filename)
  or die "can't open file $cidr_filename: $!";

print "$iptables -F $chainname\n";
print "$iptables -X $chainname\n";
print "$iptables -N $chainname\n";

while (<$cidr_fh>) {
  $line=$_;
  ($ip1,$ip2,$ip3,$ip4,$mask) = §
    /^(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})\/(\d{1,2})/;
  if ($ip1 ne "" && $ip2 ne "" && $ip3 ne "" && $ip4 ne "" && $mask ne "") {
    print "$iptables -A $chainname -s $ip1. §
      $ip2.$ip3.$ip4/$mask -j $chainrule\n";
  }
}
```

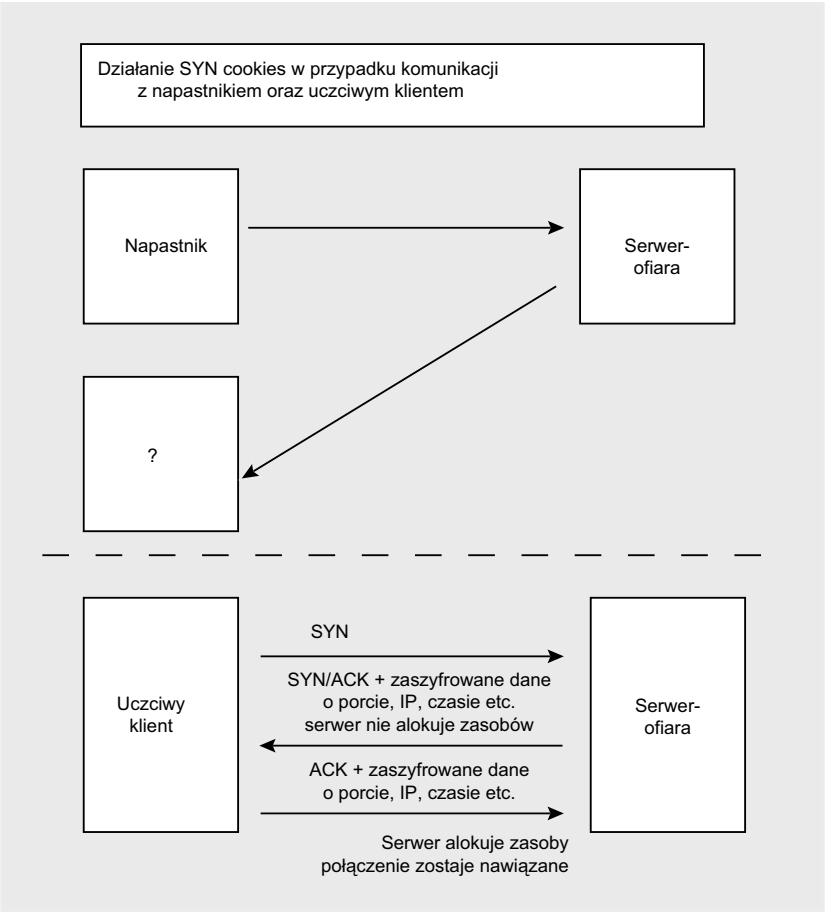
bogon-list.html udostępnia bogate listy *bogonów*. Mimo że wciąż dokonywane są nowe alokacje, ich zawartość zmienia się nieregularnie.

Sytuacja jest jeszcze bardziej skomplikowana: adresy rozpoczynające się od oktetu 192, typowe dla sieci prywatnych, mogą być poprawne w Internecie. Do znalezienia informacji o przypisaniach mniejszych bloków konieczne byłoby sprawdzenie zasobów czterech regionalnych rejestrów internetowych: ARIN, APNIC, LACNIC i RIPE NCC (więcej informacji o regionalnych rejestrach internetowych w Artykule *Jak zdemaskować nadawcę listu* – przyp. red.). Domyślać się można, że to sporo pracy. Możemy jednak liczyć na ułatwienie – gotowe, przetworzone listy. Znajdziemy je na stronie *http://www.completewhois.com/bogons/*. Nowe wersje tworzone są codziennie.

Tak jak w poprzednim przypadku, również tutaj mamy do wyboru format zapisu danych. Wykorzystując nawet najbardziej zwięzły z nich – *CIDR Bit Notation* (adres/bity maski) – lista ma objętość ponad 120 KB! Dla porównania: adresy już wykorzystane, zapisane w tej samej postaci, mają objętość nieco powyżej 260 KB.

Należałoby gotowe listy zaprząc do pracy, dokonując przekształcenia na odpowiednie wywołania *iptables*, blokujące ruch. Ręczna konwersja naturalnie nie ma sensu – autorzy listy oferują więc skrypt, który służy temu celowi (patrz Listing 4).

Kolejny skrypt pochodzący z *http://www.completewhois.com* – *bogon_iptable_update* – może zadbać o aktualizację wykorzystywanej listy, pobierając jej nową wersję.



Rysunek 2. *Zasada działania SYN cookies*

Niestety, liczba wpisów (ponad 8 tysięcy) wpłynęłaby zapewne na szybkość pracy routera lub komputera filtrującego. Administrator musi więc podjąć decyzję, czego użyć – ogólnych, ale krótkich listy z IANA lub Cymru Team, czy wyjątkowo szczegółowych, a przez to ogromnych list Completewhois.

Kontrowersyjne ciasteczka

Jedną z form obrony przeciwko atakom typu *SYN flood* może być zastosowanie tzw. *SYN cookies*. Jest to opcjonalne usprawnienie protokołu TCP, obecne w nowszych jądrach Linuksa, BSD i Solaris. Pozwala ono zminimalizować przeciążenie serwera podczas ataku.

Zasada działania jest dość prosta. Gdy w atakowanym systemie wyczerpie się miejsce w kolejce połączeń półotwartych, zostaje uruchomiony wspomniany mechanizm *SYN cookies*. Podczas zapytania kolejnego klienta o możliwość otwarcia połączenia, atakowany serwer nie tworzy już nowego połączenia półotwartego, lecz szyfruje dane na temat swojego stanu (adresy IP, porty) w ciasteczku, które jest odsyłane klientowi razem z pakietem SYN/ACK. Jeśli klient jest rzeczywistym nadawcą pakietu SYN/ACK, z powrotem przysyła ciasteczko i na jego podstawie zostaje nawiązane połączenie.

W większości dystrybucyjnych jąder mechanizm *SYN cookies* jest wkompilowany, ale nieaktywny. Trzeba go włączać (lub dodać do skryptów startowych) przy każdym uru-

W Sieci

- *http://www.securityfocus.com/infocus/1729* – artykuł na temat przystosowania stosu TCP/IP do odpierania ataków SYN,
- *http://www.completewhois.com/* – informacje o adresach i numerach IP oraz bogonach,
- *http://www.cymru.com/Bogons* – informacje o bogonach,
- *http://www.iana.org/* – Internet Assigned Numbers Authority,
- *http://cr.yp.to/syncookies.html* – witryna Daniela J. Bernsteina o *SYN cookies*.

chomieniu maszyny za pomocą polecenia:

```
echo 1 > \
  /proc/sys/net/ipv4/tcp_syncookies
```

Brak powyższego pliku w systemie oznacza konieczność wkompilowania tej funkcjonalności w kernel. Aby uzyskać możliwość korzystania z *SYN cookies*, w jądrach serii 2.4 należy zaznaczyć opcję *Networking Options -> TCP/IP Networking -> TCP syncookie support*. W jądrach serii 2.6 opcja ta zmieniła nieco swoje położenie: *Device Drivers -> Networking Support -> Networking Options -> IP: TCP syncookie support*. Następnie trzeba skompilować i zainstalować nowy kernel.

Opisywana technika wzbudziła wiele kontrowersji w środowisku komputerowym. Pojawiły się głosy, że stosowanie *SYN cookies* może bardziej szkodzić niż pomagać. Na przykład stosując *SYN cookies* nie można jednocześnie stosować niektórych rozszerzeń TCP – takich jak *large windows*, czyli wysyłanie dużej ilości danych bez potwierdzenia. Innym argumentem przeciwników jest twierdzenie, że *SYN cookies* stanowią naruszenie protokołu TCP.

Distributed Content Networks

Dobrze zaplanowany i skutecznie przeprowadzony atak DDoS może spowodować niemałe straty. Słynny robak *Blaster*, infekujący systemy Windows, atakował serwery Microsoftu. Jednak agresja ta spowodowała tylko jednodniowy przestój w funkcjonowaniu serwerów WWW korporacji. Przerwa była krótka dlatego, że Microsoft zdecydował się na przeniesienie swojej powitalnej witryny oraz części serwerów DNS do tzw. *Distributed Content Network* (sieci rozproszonej zawartości). Technika ta umożliwia umieszczenie danych, na przykład strony WWW, na dużej liczbie maszyn jednocześnie, co pozwala rozłożyć atak na mniejsze części. W ten sposób pojedyncze komputery są w stanie poradzić sobie ze zwielokrotnionym natężeniem ruchu sieciowego.

Ta sama technika mogła ograniczyć skutki ataku tegorocznego robaka *MyDoom* na firmę SCO Group, która zasłynęła roszczeniami wobec Linuksa (SCO uważa, że kod źródłowy jądra narusza jej prawa patentowe i żąda od użytkowników tego systemu sporych opłat licencyjnych). Jednak serwery SCO nie miały szans na oparcie się atakowi. Skorzystanie z usług dowolnej z trzech największych sieci DCN – wszystkie używają Linuksa – byłoby jednoznaczne z koniecznością wystawienia takiej firmie 699$ rachunku za każdą kopię Linuksa, którą ona posiada. Naturalnie takie rozwiązanie było nierealne. Witryna *http://www.sco.com* zniknęła na pewien czas z sieci.

Technika DCN, choć bardzo skuteczna, ma jedną wadę – jest droga. Mogą sobie na nią pozwolić jedynie duże firmy. Próba stworzenia tego typu sieci we własnym zakresie – nawet gdyby się powiodła – mogłaby się okazać zbyt kosztowna w eksploatacji. ■